Description of the DLL regulation interface in HAWC

Larsen, Torben J.

Publication date:
2001

Document Version
Publisher's PDF, also known as Version of record

Risø-R-1290(EN)

Description of the DLL Regulation Interface in HAWC

Torben J. Larsen

Risø National Laboratory, Roskilde, Denmark
September 2001

Abstract This report contains a description of the external regulation interface between the aeroelastic code HAWC and a seperate regulation unit programmed as a DLL (**D**ynamic **L**ink **L**ibrary). Specific HAWC commands used with the regulation as well as simple DLL examples written in Delphi, Fortran and C are included in the report.

The work was funded by the Danish Energy Agency in the contract ENS 1363/01-0001.

ISBN 87-550-2933-7
ISBN 87-550-2934-5 (Internet)
ISSN 0106-2840

Print: Pitney Bowes Management Services Denmark A/S, 2001.

Contents

1 Introduction

For the aeroelastic calculations of wind turbines today the regulation strategy plays an important role. The versions of regulations change rapidly as they are continuously improved and new strategies are tested together with aeroelastic load calculations. This together with the fact that many users of aeroelastic codes mainly wish to consider the regulation as a "black box" and that the regulator in many cases contains confidential information in the code, or is written in another language than the aeroelastic code, has motivated the work of implementation of external regulation units.

The method of implementing the external regulation is to compile it as an independent unit, a DLL (**D**ynamic **L**ink **L**ibrary). This method is mostly used in the windows environment to share small programs between larger programs, but can also be used for other purposes.

1.1 Main overview

The main idea of the DLL is to divide the control unit into a few procedures that can be called by HAWC. The first procedure is the regulator, which basically consist of the controlling algorithm of the pitch and power references. The second procedure is the pitch servo algorithm, that calculates the transient position of the pitch angles between the pitch angle set points. The last procedure is the generator subroutine which calculates the generator mechanical torque transient corresponding to the rotational speed of the generator and the reference electrical power. This enables variable speed control through active power setting.

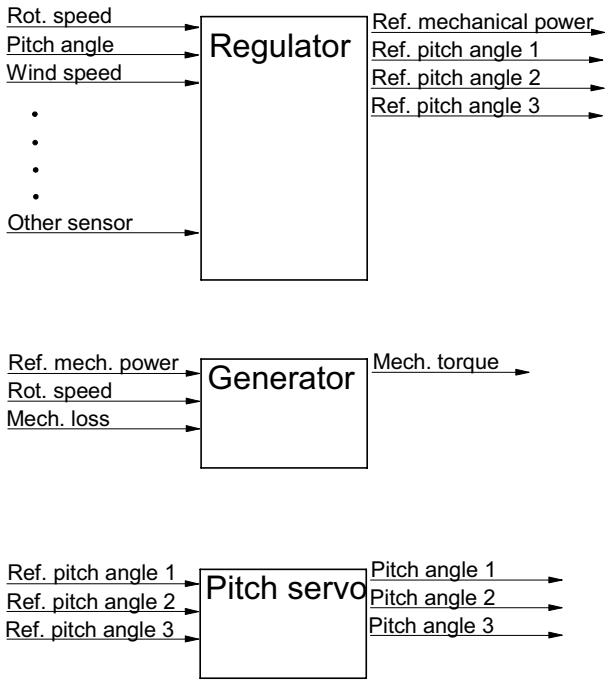

Figure 1. Principle of subroutine calls in DLL.

Since the regulator is normally called with a certain sample frequency this procedure is called with a different frequency than the two other procedures. Some regulators operates with a sample frequency of 10[Hz], where as the aeroelastic code HAWC normally operates with a time step corresponding to 32-50[Hz] depending on the turbine size and type. The two procedures *generator* and *pitchservo* operates with the same frequency as the aeroelastic code since they calculate the instantaneous values of respectively generator torque and pitch angles.

The DLL have to placed in the HAWC DATA directory together with the _st files etc.

2 Commands

Several commands in HAWC has to be used to make the DLL call working. These procedures are in the following text written in *italics* (*regulation, pitchservo, generator*).

Together with the following commands some of the basic HAWC commands has to be used to make the regulation operate.

1. PARAMETERS TURBINE Parameter number 3 (Variable speed) has to be set to 4 to indicate a controlled variable speed.
2. ROTOR BASIC_LAYOUT has to be specified to ensure that a pitch bearing is defined.
3. PARAMETERS PITCHABLE_BLADE_PART has to be specified, also to ensure the pitch bearing. Pitch offset relative to the pitch controller is also set here.

HawC commands related to the external DLL interface

Command	Key word	Position words
DEFINE_STRING	EXTERNAL_REGULATION-_NAME	Specify the name (with extension) of the external DLL control file in the DATA directory.
PARAMETERS	EXTERNAL_REGULATION	1. Time increment between call to procedure *regulation* in external DLL. It has to be a multiple of the HAWC time increment defined in the command WRITE RESPONSE
EXT_CONTROL-_SENSOR	AZIMUTH	Place azimuth angle in the *regulation* array to the external DLL. 1. Azimuth offset [°]. The reference is blade 1, and the azimuth position for this blade is zero, when pointing vertically downwards. 2. Position of azimuth sensor in *regulation* array to the external DLL.
EXT_CONTROL-_SENSOR	BLADE_FORCE-_MOMENT	Forces and moments at blade nodes are placed in the *regulation* array to the external DLL. 1. Blade number. 2. Node number. The following 6 parameters correspond to forces (the first 3) and moments (the next 3). A force or moment is placed in the *regulation* array if the parameter is 1 and neglected if the parameter is 0: 3. F_x. 4. F_y. 5. F_z. 6. M_x. 7. M_y. 8. M_z. The following 6 parameters indicates the position in the *regulation* array. If they are not chosen in the previous 6 parameters the number is neglected. 9. Position of F_x. 10. Position of F_y. 11. Position of F_z. 12. Position of M_x. 13. Position of M_y. 14. Position of M_z.
EXT_CONTROL-_SENSOR	GENERATOR_SPEED	The rotational speed of the generator node times the gear ratio is written to the *regulation* array. 1. Position of the generator speed in the *regulation* array

continued on next page ...

. . . continued from previous page

Command	Key word	Position words
EXT_CONTROL- _SENSOR	PITCH_BEARING_ANGLE	A pitch angle is placed in the *regulation* array if the parameter is 1 and neglected if the parameter is 0. 1. Pitch angle of bearing blade 1 2. Pitch angle of bearing blade 2 3. Pitch angle of bearing blade 3 The following 3 parameters indicates the position in the *regulation* array. If they are not chosen in the previous 3 parameters the number is neglected. 4. Position of pitch angle of bearing blade 1 5. Position of pitch angle of bearing blade 2 6. Position of pitch angle of bearing blade 3
EXT_CONTROL- _SENSOR	TOWER_ACCELERATION	1. Tower node The following 6 parameters corresponds to tower acceleration. A tower acceleration is placed in the *regulation* array if the parameter is 1 and neglected if the parameter is 0. 2. $\ddot{x}$ 3. $\ddot{y}$ 4. $\ddot{z}$ 5. $\ddot{\theta}_x$ 6. $\ddot{\theta}_y$ 7. $\ddot{\theta}_z z$ The following 6 parameters indicates the position in the *regulation* array. If they are not chosen in the previous 6 parameters the number is neglected. 8. Position of $\ddot{x}$ 9. Position of $\ddot{y}$ 10. Position of $\ddot{z}$ 11. Position of $\ddot{\theta}_x$ 12. Position of $\ddot{\theta}_y$ 13. Position of $\ddot{\theta}_z$
EXT_CONTROL- _SENSOR	WIND_SPEED_AT_HUB	The three components of of the wind can be chosen in the following 4 parameters. If the parameter is 1 the corresponding sensor is placed in the *regulation* array, and if the parameter is 0 the sensor is neglected. 1. v_x component, transverse 2. v_y component, longitudinal 3. v_z component, vertical 4. The horizontal component, i.e. $\sqrt{v_x^2 + v_y^2}$. This corresponds to a cup-anemometer placed in the rotor center The following 4 parameters indicates the position in the *regulation* array. If they are not chosen in the previous 4 parameters the number is neglected. 5. Position of v_x 6. Position of v_y 7. Position of v_z 8. Position of the horizontal component

3 DLL format

3.1 Demands from HAWC

Procedure regulation

As sketched in section 1.1 and figure 1, three subroutines in the DLL are called from the HAWC core. At every regulation time step (specified with the command PARAMETERS EXTERNAL_REGULATION) the procedure *regulation* is called in the DLL. The procedure call from HAWC is as follows

```
CALL REGULATION(OUTVEC_X,INPVEC_X)
```

where OUTVEC_X is a one dimensional array with 15 elements of the type REAL*4. The array consist of the outgoing variables from the HAWC core to the external DLL. The content of the elements is specified with the parameters EXT_CONTROL_SENSOR.

The array INPVEC_X is a one dimensional array with 4 elements of the type REAL*4. The array consist of the incoming variables from the external DLL to the HAWC core. The content is always

`INPVEC_X[1]` : Reference power from the regulator [W].
`INPVEC_X[2]` : Reference pitch angle of blade 1 [rad].
`INPVEC_X[3]` : Reference pitch angle of blade 2 [rad].
`INPVEC_X[4]` : Reference pitch angle of blade 3 [rad].

The sign convention of the pitch angles is according to the HAWC definition, which means that a pitch controlled wind turbine have high negative pitch angle at high wind speed. The blade numbering is also according to the HAWC definition, which means that a turbine seen from the wind has blade 2 at 10 o'clock and blade 3 at 2 o'clock when blade 1 is at 6 o'clock.

Procedure pichservo

The procedure *pitchservo* is called at every time step in the HAWC calculation. The procedure call from HAWC is as follows

```
CALL PITCHSERVO(TIMSTP_X,PITREF_X,PITANG_X)
```

where `TIMSTP_X` is a REAL*4 variable, that contains the time increment size of the HAWC calculation defined with the command WRITE RESPONSE.

The array PITREF_X is a one dimensional array with 3 REAL*4 elements. The array consist of the reference pitch angles calculated in the REGULATION procedure. The content is always

`PITREF_X[1]` : Reference pitch angle of blade 1 [rad].
`PITREF_X[2]` : Reference pitch angle of blade 2 [rad].
`PITREF_X[3]` : Reference pitch angle of blade 3 [rad].

The array PITANG_X is a one dimensional array with 3 REAL*4 elements. The array consist of the actual pitch angles calculated in the PITCHSERVO procedure. The content is always

`PITANG_X[1]` : Pitch angle of blade 1 [rad].
`PITANG_X[2]` : Pitch angle of blade 2 [rad].
`PITANG_X[3]` : Pitch angle of blade 3 [rad].

Procedure generator

The procedure *generator* is called at every time step in the HAWC calculation. The procedure call from HAWC is as follows

```
CALL GENERATOR(TIMSTP_X,POEREF_X,POWLOS_X,OGAGEN_X,MGENERA_X)
```

where `TIMSTP_X` is a REAL*4 variable, that contains the time increment size of the HAWC calculation defined with the command WRITE RESPONSE.

`POEREF_X` is a REAL*4 variable containing the reference power [W] from the REGULATION procedure.

`POWLOS_X` is a REAL*4 variable that contains the mechanical and electrical power loss [W] as specified in that HAWC_ GE and HAWC_ GB files.

`OGAGEN_X` is a REAL*4 variable that contains the generator speed [rad/s] (rotor speed at the first nacelle node times the gear ratio).

`MGENERA_X` is a REAL*4 variable that contains the generator torque [Nm] on the high speed side of the gear box as calculated in the procedure GENERATOR.

3.2 Example of DLL syntax written in Delphi, ref. [7]

```
library control;
uses  SysUtils;
Type
  array_15   = array[1..15] of single;
  array_3    = array[1..3]  of single;
  array_4    = array[1..4]  of single;

Procedure regulation(var inputvektor: array_15;
                     var ouputvektor: array_4);
stdcall;
var
begin

end;


Procedure generator(var timeinc,powref,mektab,omegagen,mgenera : single);
stdcall;
var
begin

end;


Procedure pitchservo(var timeinc : single;
                     var thetaref, theta : array_3);
stdcall;
var
begin

end;

exports
  regulation, generator, pitchservo;

end.
```

3.3 Example of DLL syntax written in Fortran, ref. [6]

```
      SUBROUTINE REGULATION(INPUTVEKTOR,OUTPUTVEKTOR)
      IMPLICIT NONE
*
      REAL*4 INPUTVEKTOR(15),
     +       OUTPUTVEKTOR(4)
*
!DEC$ ATTRIBUTES DLLEXPORT::REGULATION
*
*  Procedure calculations ... *
*
      RETURN
      END
*
*-------------------------------------------------------------*
      SUBROUTINE PITCHSERVO(TIMSTP,PITREF,PITANG)
      IMPLICIT NONE
*
      REAL*4 TIMSTP,
     +       PITREF(4),
     +       PITANG(4)
*
!DEC$ ATTRIBUTES DLLEXPORT::PITCHSERVO
*
*  Procedure calculations ... *
*
      RETURN
      END
*
*-------------------------------------------------------------*
      SUBROUTINE GENERATOR(TIMSTP,POEREF,POWLOS,OGAGEN,MGENERA)
      IMPLICIT NONE
*
      REAL*4 TIMSTP,
     +       POEREF,
     +       POWLOS,
     +       OGAGEN,
     +       MGENERA
*
!DEC$ ATTRIBUTES DLLEXPORT::GENERATOR
*
*  Procedure calculations ... *
*
      RETURN
      END
```

3.4 Example of DLL syntax written in C, ref. [8]

```
#define DLL_EXPORT __declspec( dllexport )

DLL_EXPORT void regulation(float inputvektor[15], float outputvektor[4])
{
    /* Procedure calculations */
}

/*-------------------------------------------------------------------------*/

DLL_EXPORT void pitchservo(float *timstp, float pitref[4], float pitang[4])
{
    /* Procedure calculations */
}

/*-------------------------------------------------------------------------*/

DLL_EXPORT void generator(float *timstp,float *poeref,float *powlos,float*ogagen,float *mgenera)
{
    /* Procedure calculations */
}
```

4 HAWC coordinates

The HAWC coordinate system regarding the external DLL interface is divided into 3 subsections tower, shaft and blades.

The tower coordinate system is fixed and illustrated in figure 2 as the x_T, y_T, z_T system.

The shaft coordinate system is rotating with the rotor and illustrated in figure 2 as the x_A, y_A, z_A system.

The blade coordinates follows the local principle bending axes a long the blades, which normally is the same as the local chord coordinate system. This affects that a turbine with pitching blades have a non pitching coordinate system in the first blade nodes and a pitching coordinate system from the bearing node to the tip.

The blade coordinate system is illustrated in figure 2. The coordinate system is the x_L, y_L, z_L system until the blade bearing (including the blade bearing). From the blade bearing the coordinate system is illustrated in figure 2 as the x_P, y_P, z_P system.

It should be noticed that the blade coordinate system results in an opposite sign convention for the pitch angle than normally used in other wind turbine coordinate systems.

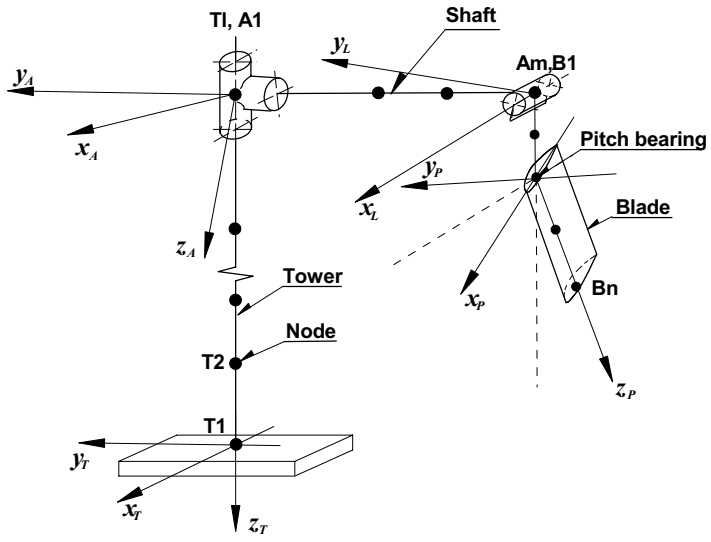

Figure 2. Coordinates systems used in HAWC

References

[1] Petersen, J. T. (2001) HAWC - Wind Turbine Simulation Code - User's Guide. Risø-I-1408(en)

[2] Petersen, J. T. (1996) The Aeroelastic Code HawC - Model and Comparisons. In proc. of the 28th IEA Expert Meeting 'state of the art of aeroelastic codes'. Lyngby, Danmark.

[3] Petersen, J.T. *Kinematically Nonlinear Finite Element Model of a Horizontal Axis Wind Turbine.*
Ph.D. thesis, Part 1: Mathematical Model and Results.
Dept. of Meteorology and Wind Energy.
Risø National Laboratory. Roskilde, Denmark, 1990.

[4] Petersen, J.T. *Kinematically Nonlinear Finite Element Model of a Horizontal Axis Wind Turbine.*
Ph.D. thesis, Part 2: Supplement. Inertia Matrices and Aerodynamic Model
Dept. of Meteorology and Wind Energy.
Risø National Laboratory. Roskilde, Denmark, 1990.

[5] *Lahey Fortran 90 User's Guide.* 1995-7 Lahey Computer Systems Inc.

[6] *Compaq Visual Fortran 6.1* 1997-1999 Digital Equipment Corporation.

[7] *Borland Delphi 5.0.* 1983, 1999 Inprise Corporation.

[8] *Borland C++ 5.02* 1991, 1997 Borland International Inc.

Bibliographic Data Sheet **Risø-R-1290(EN)**

Title and authors

Description of the DLL regulation interface in HAWC

Torben J. Larsen

ISBN	ISSN
87-550-2933-7; 87-550-2934-5 (Internet)	0106-2840

Department or group	Date
Aeroelastic Design Wind Energy Department	Sept. 2001

Groups own reg. number(s)	Project/contract No(s)
1110029-00	ENS-1363/01-0001

Sponsorship

Pages	Tables	Illustrations	References
16	1	2	8

Abstract (max. 2000 characters)

This report contains a description of the external regulation interface between the aeroelastic code HAWC and a separate regulation unit programmed as a DLL (**D**ynamic **L**ink **L**ibrary). Specific HAWC commands used with the regulation as well as simple DLL examples written in Delphi, Fortran and C are included in the report.

Descriptors INIS/EDB

Available on request from Information Service Department, Risø National Laboratory,
(Afdelingen for Informationsservice, Forskningscenter Risø), P.O.Box 49, DK-4000 Roskilde, Denmark.
Telephone +45 4677 4004, Telefax +45 4677 4013, E-mail: risoe@risoe.dk